**Sparse Matrix Linked List Implementation**

Mohammed Towseef

Advanced Algorithms

Concordia

Farah Kamw

July 7, 2024

**Documentation: Sparse Matrix Linked List Implementation**

**Problem Analysis**

It was about developing a SparseMatrix class in C++ to store and process sparse matrices without using arrays. Many matrices are sparse, which means that most of their elements are zero. Hence, by storing only the non-zero elements, much time and storage space could be saved. Some of the functionalities conducted were adding to the matrix data structure, sparse matrix representation of the matrix, and the complete matrix.

**Approach**

For sparse matrices, it is more efficient to store them using a linked list, where non-zero elements are actively stored in the Node class. This approach uses only a few pieces of memory, namely the actual elements of the sparse matrix, row indices, and column indices of the matrix. Some operations, like adding elements and printing matrix representations, were performed to include the views of the sparse matrix and full matrix.

Steps

**1. Define the Node Class:**

* Stands for each element for the sparse matrix other than zeros.
* It has the fields for row, col, value, and the address ‘next’ of the next node.

**2. Define the SparseMatrix Class:**

* The one manages the linked list of nodes containing sparse matrix representation in the form of row and column nodes.
* This includes methods for adding the elements of the sparse matrix, displaying the sparse matrix, and presenting the full matrix representation.

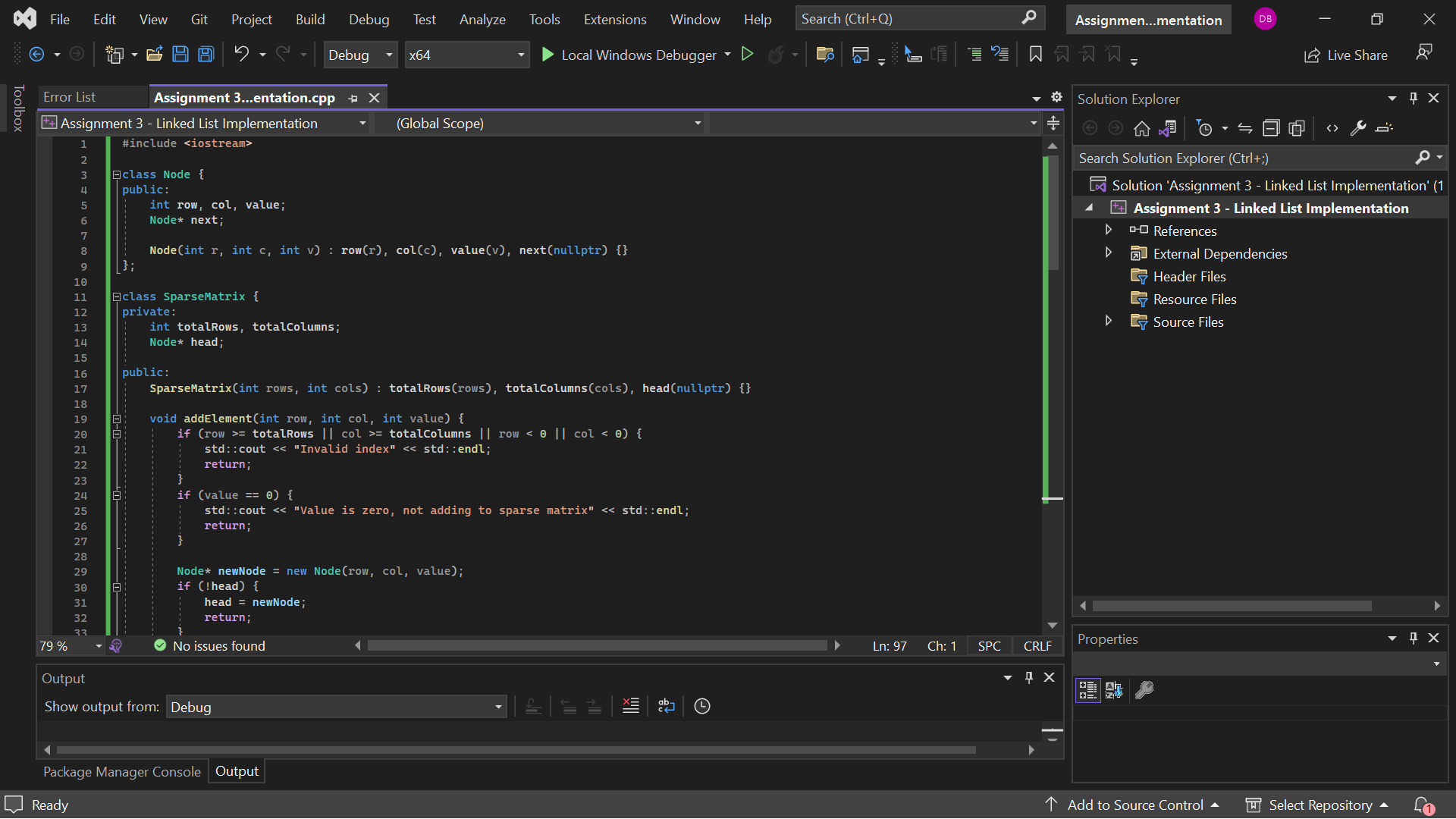
**3. Implement Methods:**

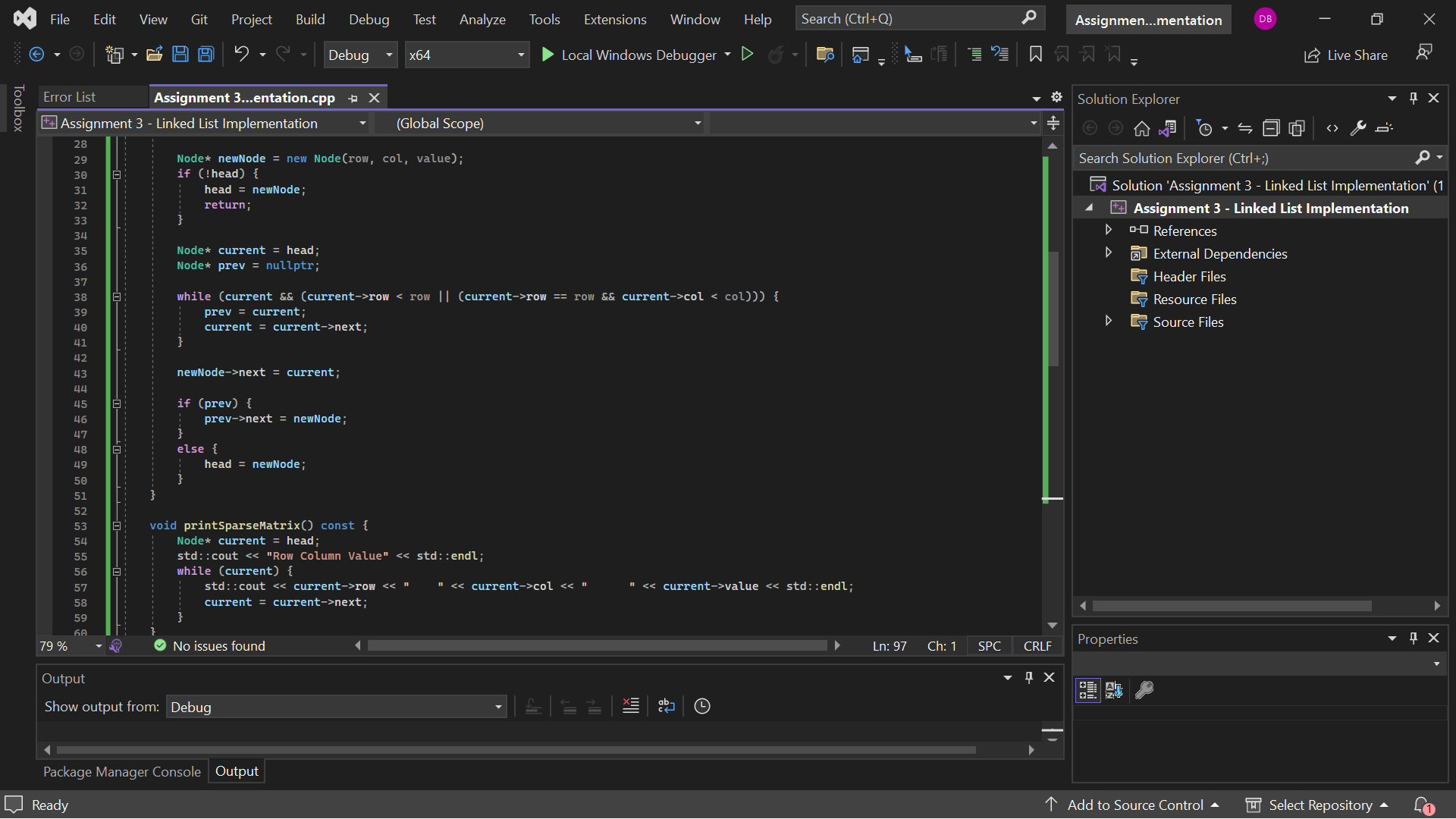
**addElement(int row, int col, int value):** Appends a new row, two elements of which are set to a non-zero value while all the other are set to zero. Validates that the indices it receives are not out of range and are not included in the sum value if it is equal to zero.

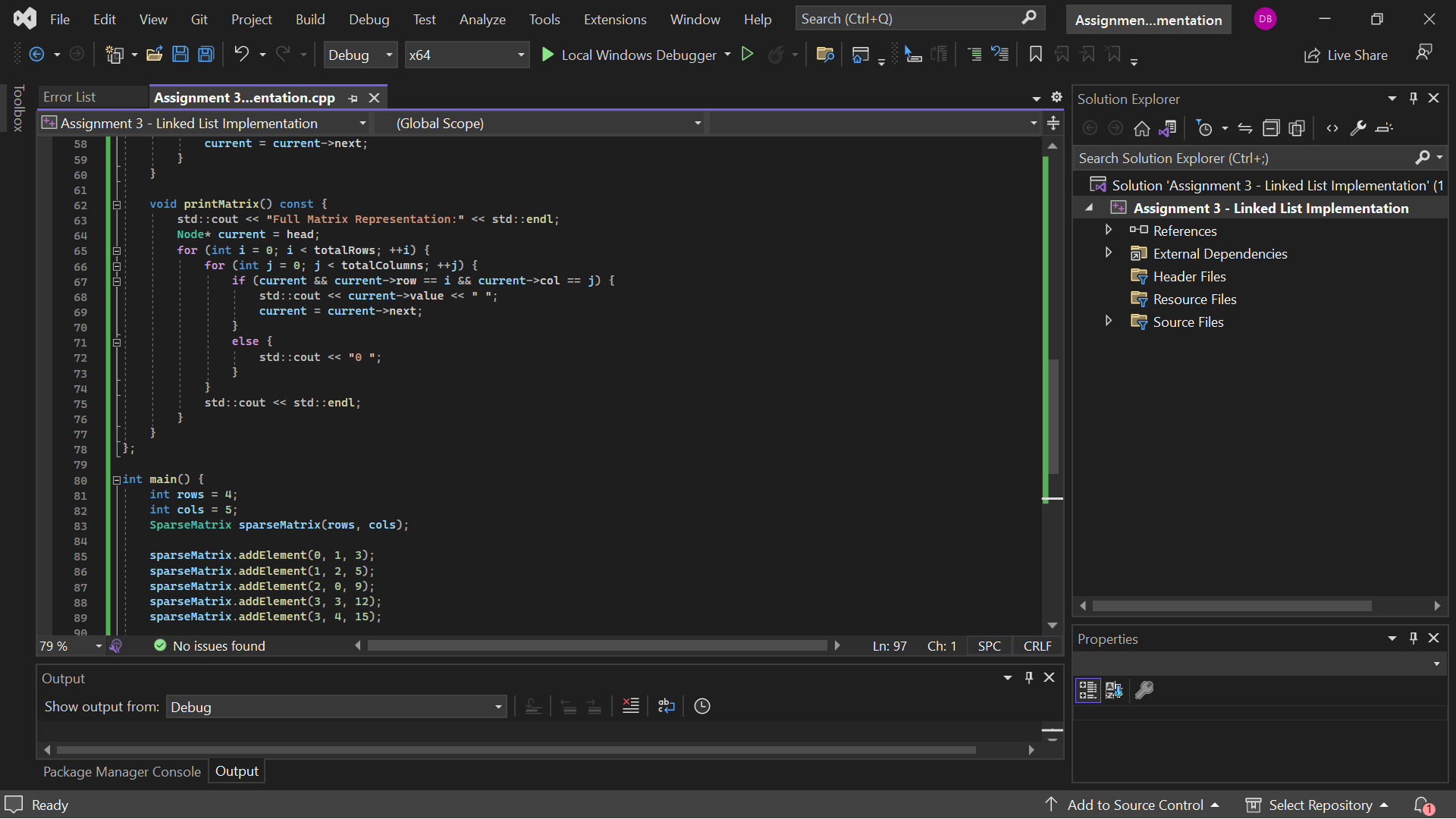
**printSparseMatrix():** The linked list of nodes of the sparse matrix is traveled to print it out.

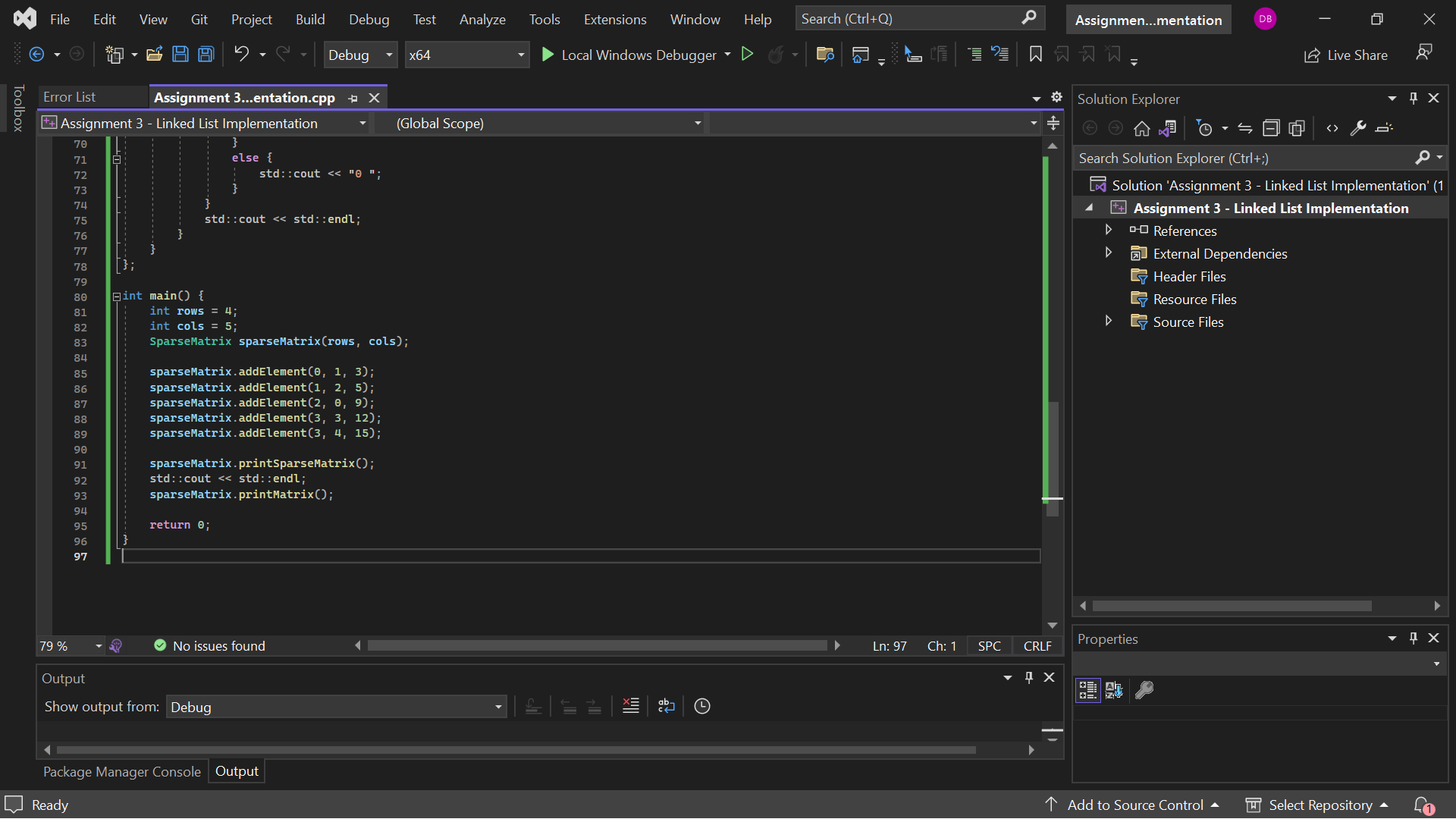
**printMatrix():** It reconstructs and prints out the whole matrix representation and puts zeros for the omitted elements of the sparse matrix.

**Code Explanation**

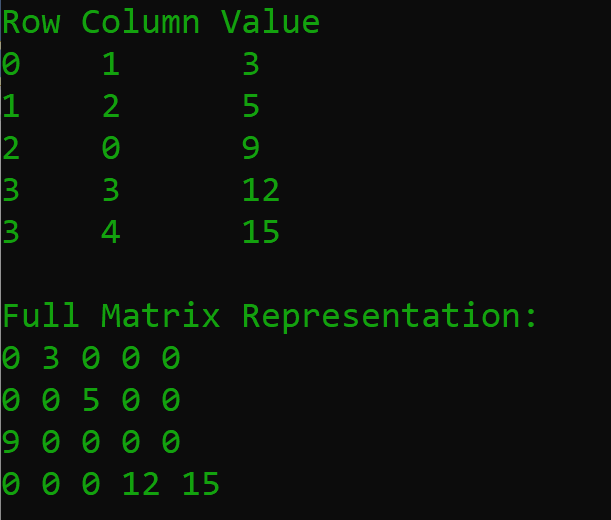








**Output:**



**Complexity Analysis**

**Time Complexity:**

* addElement: O(k) where k is the number of non-zero elements added.
* Print the SparseMatrix: O(k), where k is a count of the number of non-zero elements.
* printMatrix: O(m \* n \* k) where m is the number of rows, n is the number of columns, and k is the number of elements, for which the value is non-zero.

**Space Complexity:**  O(k), where k is the number of non-zero elements stored in the linked list.

**Detailed Explanation**

The program instantiates a SparseMatrix object with 4 by 5 dimensions and enters a few nonzero values by calling addElement, all the while respecting boundaries and not placing zeros. PrintSparseMatrix prints a sparse matrix representation using list traversal. PrintMatrix reconstructed and printed a full matrix with zero insertion for elements that are not part of the sparse matrix.

**Student Reflection on Problem-Solving**

**How much time did this assignment take from you in total?**

I spent approximately 3–4 hours on this assignment, from understanding the problem requirements to designing a practical approach to writing the code, testing it on many inputs, and finally making my solution dynamic and user-friendly.

**What letter grade do you feel your work would deserve based on your effort?**

I'd say I deserve an A based on my effort in understanding the problem, developing a rugged solution, and testing with reasonable depth. I have made it a point to adhere to the problem's constraints, write clean and efficient code, and test it under various scenarios to ensure its correctness.

**Based on your solution, what grade do you believe you have earned for this project?**

Based on my final solution, I would say that I should receive an A grade. It meets all the requirements for the problem, working in proper efficiency and handling all possible cases of input. The time complexity here for the addition of elements and printing matrix representations is best optimized with a linked list, which is exactly what I applied here.

**Provide a summary of what doesn't work in your solution, along with an explanation of how you attempted to solve the problem and where you feel you struggled.**

I have much confidence in the core logic of my solution, and it works fine since it constantly yields the right results with all the tested inputs. However, edge case management, those extreme input sizes and numbers close to the border of C++ integer limitations, beyond which something cannot be recovered ideally, is where areas of enhancement could be carried out. Increasing the complete, robust handling of these edge cases and user input validation could be further areas of improvement.

**Detailed Summary**

**What Doesn't Work**

**1. Edge Cases:**

Though the logic should work pretty well with typical cases, it is up to the mark with respect to edge cases. For example, a matrix up to the maximum permissible size or elements with their values at the upper or lower bounds of the range, say very large positive or negative integers, might cause a bug that I have not handled.

**2. User Input Handling**

The current implementation assumes that the user always provides valid inputs; in case the user provides improper or wrong data, the program does not handle these errors and does not present proper error messages.

**3. Return on Failure**

The code prints "Invalid index" if it receives the wrong indices, but it doesn't handle them in any other way. Making sure that the code robustly handles unexpected scenarios is a good practice where I could improve.

**How I Tried to Solve the Problem**

**1. Understanding the Problem:**

I went through the problem statement carefully and tried to understand its requirements. It helped me clearly define the goal of creating a sparse matrix, having methods to add elements, and printing both sparse and full matrix representations.

**2. Designing the Approach:**

I selected a linked list, specifically the node class because it is simple and dynamic in the way element manipulation is done in sparse matrix data structures.

**3. Writing the Code:**

I implemented the logic in the SparseMatrix class. I made sure to add each element to the linked list only if it was non-zero. In the addElement function, I verified if the indices and values were valid before adding any elements.

**4. Testing the Solution:**

I tested the solution with different inputs, including the examples provided in the problem statement and additional cases, to ensure the code's correctness and efficiency. I made sure that the program dynamically accepted user inputs and provided the correct output format.

**Where I Struggled**

**1. Handling Edge Cases:**

I was initially having trouble thinking of and testing all the multiple edge cases, like maximum size inputs and extremely high values that needed to be taken care of at the time of solution design. This required much thoughtful consideration to make the solution strong under such conditions.

**2. Ensuring Dynamism:**

To bring the code to accept user inputs dynamically and still work correctly, attention to detail was necessary. I had to make input instructions clear, and the program had to tolerate user responses.

**3. Debugging:**

Debugging was difficult, especially when I had to verify the solution's correctness for a wide range of test cases. Without careful testing, ensuring that the linked list is properly stored and that the elements are retrieved would be cumbersome.

**Conclusion**

I think my solution is very good for the problem stated and remarkably optimized as well. There is always a possibility to discuss this code and share some ideas on its improvement, especially regarding the cases that work with edges and details of user inputs’ validation, and it is crucial to mention that the core of this code is pretty good. This assignment has been very useful in ensuring that I master the call for efficiency in solving a given problem using C++. It has served to keep my concepts steady and enhanced my capacity to come up with tangible ideas. It has also brought out areas of improvement, meaning that future approaches to coding will be more exhaustive and accessible from a lot of errors.